Setting up a Raspberry Pi

As there is a Raspberry Pi inside the PiDP, there are endless configuration options you can tinker with. Which is fun, flexible, and highly confusing. Start with a **simple** setup, and add any desired confusion later on.
The PiDP can happily run as a PDP-8, and at the same time do anything else you'd like a Pi to do. Use it **concurrently** as a file server, media server or home automation system. Please test your Pi with the PiDP-8 software **before** you build the kit, so you know the software bit is good before you start soldering. The PiDP software runs fine without the PiDP hardware.

The basics

Download the normal Raspbian SD card image from www.raspberrypi.org/downloads/raspbian/ . Choose the full version with GUI (you'll need an 8 or better, a 16GB SD card for all PDP-8 software), as the PiDP uses X in some situations. You can use a program called Etcher (www.balena.io/etcher/) to put the downloaded image file on your SD card. Remember: the default login is pi, with password raspberry.

A. If you use a HDMI monitor and USB keyboard:

1. Insert the SD card into the Pi and power up. Set up Wifi. You can either do that in the GUI, or using 'sudo raspi-config', whatever you prefer. Detailed instructions are on www.digikey.com/en/maker/blogs/raspberry-pi-3---how-to-connect-wi-fi-and-bluetooth.

2. Enable ssh and VNC. In the GUI: Raspberry Icon->Preferences->Raspberry Pi Configuration->Interfaces tab. In raspi-config: main menu, Interfacing Options. Don't enable Bluetooth unless you will not be using the built-in serial port of the PiDP.

B. If you do *not* have a HDMI monitor to plug in to the Pi: you can also set up wifi and ssh on the SD card before you even put it in your Pi. Then you can use ssh sessions remotely to do everything. Using VNC you can even use the GUI. There is really no need for a HDMI monitor. Three steps:

1. Insert the SD card into your PC, and on the boot partition (which a PC can see) create and empty file named 'ssh'.

2. Create a file 'wpa_supplicant.conf' and use a text editor to put the following in it (edit the ssid name and psk password to match your own wifi, and change the country code if you care):

```
country=us
update_config=1
ctrl_interface=/var/run/wpa_supplicant

network={
 scan_ssid=1
 ssid="MySSID"
 psk="MyPassword"
}
```

3. Insert the SD card into your Pi, boot, and see the following chapter to connect wirelessly.

*Important: **make sure to use a good power supply**. The newer the Pi, the more sensitive it is to even the slightest power glitch, leading to crashes and corrupted SD cards much more often than you'd think. Use the Official Raspberry Pi Power Supply if in doubt, it's only $12.50 or so. Note the Pi likes 5.1-5.35V, a 5.00V power supply is only marginally OK! Lastly, if you decide to power the PiDP through the key switch, and you use the Pi 3B+, read the Hardware Notes at the end of this manual.*

Wireless operation

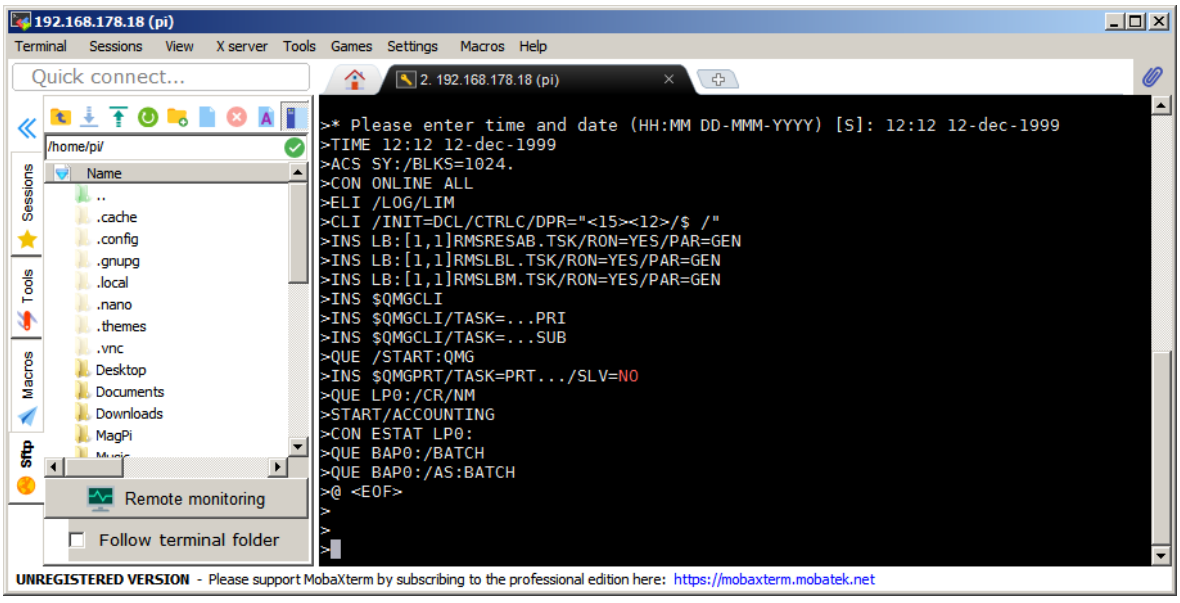
This section is not about the PiDP, just about the Pi itself. Many people do not realise how tightly the Pi can be integrated with your laptop. It makes using the Pi (and PiDP) much more pleasant. Place the PiDP anywhere and use it wirelessly from the sofa.

I greatly recommend an all-in-one Windows program called **MobaXterm** (mobaxterm.mobatek.net). Linux has its own tools, and other Windows programs are available (puTTY, Tera Term). But if you are new to the Pi, start with Moba.

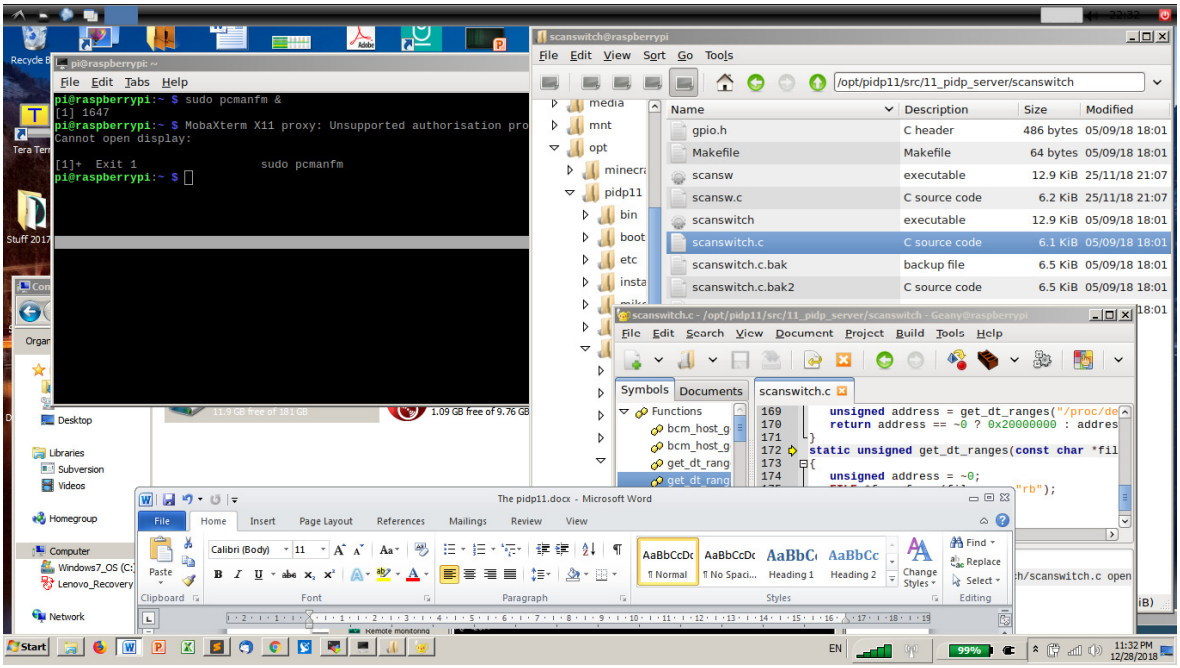
First, you will need to **find out the IP address of the Pi**. Use the Tools->Network Scanner in Moba to find that out, you'll see the Pi appear in the scanner's list (sometimes you have to repeat the scan a few times). It's the entry with ssh enabled, if you do not see its name. Click on the Deep Scan icon on that line and note the **port number**.

Now you can use Sessions->New Session to **set up an SSH terminal into your Pi**. Click on the SSH icon and enter the Pi's IP number. Check if the port number is right. You can also enter your user ID ('pi', normally) for added comfort.

To connect, double-click the newly created User Session (top right of the Moba window). You should be in. Also, you will see a file browser to the left of the terminal. You can **drag and drop files between your laptop and your Pi**, but of course you cannot save files in every directory of the Pi without proper permissions set up. Right-click text files in the file browser to edit them comfortably on your PC.



Playing with Moba some more will make you see that a GUI program on the Pi can easily run as a window on your laptop. Enter `lxsession&` in your ssh command line, and you'll get **Raspberry Pi GUI programs on your laptop mixed among your Windows applications**. See below:



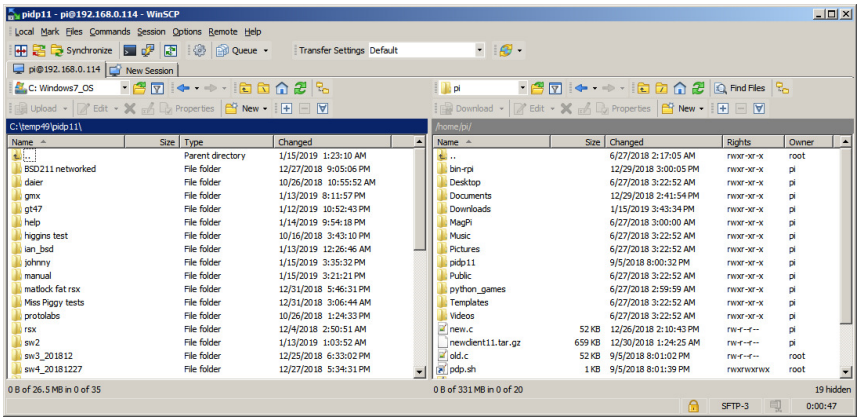
In the above picture, the Pi menu bar is set at the top, so you start Pi applications from above and Windows applications from below. Right-click that black menu bar to change its location. Perhaps a 30% sized Pi menu bar at the center bottom of the screen is better. Also, note that the window with the Pi's desktop can just be dragged off-screen. Pi programs move independent of it on your Windows desktop anyway.

For PiDP purposes, if you don't care too much about security etiquette, the command **`gksudo pcmanfm&`** is useful. It allows you to edit and move files as superuser in the pidp11 directory, making it easy to tweak the PiDP setup (with the acceptable risk of breaking something).

See www.raspberrypi-spy.co.uk/2018/12/remote-access-pi-using-mobaxterm/ for more detailed instructions on Moba.

You can also use **VNC** to get the standard Pi desktop onto your laptop. This is also the way to get any graphical output from the PDP-8 vector display (forthcoming, see Google Group) over a wireless connection. You should use Real VNC Viewer (www.realvnc.com/en/connect/download/viewer/) rather than Moba. Set the resolution/display size to something acceptable from the Pi desktop (Raspberry icon ->Preferences-> Raspberry Pi Configuration->Set Resolution button).

Another very useful, Norton Commander-style tool to move files between your laptop and



your Pi is **WinSCP**, see the picture to the right. You can download it at winscp.net .

From this point on, you will find that using the Pi remotely is simple and painless. Not strictly necessary for use as purely a PDP-11, but you'll find it useful.

Using screen, a quick walkthrough

The idea: screen is a program that acts like a virtual terminal. It is used in the PiDP setup. You can log in to it, detach from it, run programs in it. Why is that useful for the PiDP?

- You can pop in and out of the PDP-8 terminal to do stuff on the Pi, the PDP-8 just keeps going in the background.
- The PDP-8 terminal can be started on the Pi's HDMI display. Then, later on, grabbed on your laptop over ssh. Or on an xterm display. All the time, the PDP-8 just keeps running and does not care.

Ctrl-a d detaches you from screen's virtual terminal and returns you to the Linux command line. Now, re-connect to your screen by a screen –r . screen –D –r reconnects you even if screen itself objects to it. For instance, because another terminal is still connected. Now you're back in screen, do the following:

Ctrl-a ESC: brings you into the scrollback buffer. Use the cursor keys to scroll.
You can copy and paste in this mode: <space> to start marking text, <return> to copy the marked area. ESC to leave this mode. Then, CTRL-a] will paste into your prompt. Or, CTRL-a > will save the clipping to a file. Handy too: Ctrl-a < lets you paste text from some file into the terminal.

Ctrl-a | splits your screen vertically. And Ctrl-a S splits horizontally. Ctrl-a TAB jumps across splits, and in the new 'empty' slot, Ctrl-a c will create a new session. Now, you have your PDP-8 on the left, and your Pi on the right! Ctrl-a X closes the *split*, but that *session* still lives on in the background. Do another Ctrl-a c, just to create another new session. Now, to see a listing of all three screen sessions, do a Ctrl-a " , and you can select which session you want to go into. Ctrl-a <number> is a quick way of popping in and out of sessions. To end a session, type exit if the session happens to be a Linux terminal session. Or Ctrl-a k to kill and close the session, which is a bit rude. Even ruder is Ctrl-a \ , which kills everything running inside screen, and screen itself.

Screen also has a command line of its own. Ctrl-a : to enter it. Type dinfo<return> after that and screen will tell you what type of terminal you are using. But you can do all sorts of things in the screen command line.

Need to send Ctrl-a to the PDP-8? Screen is intercepting it. But Ctrl-a a will send the PDP-8 a regular Ctrl-a. Lastly, the man page on screen is very useful – even for people who do not like man pages normally!

Adding USB-serial cables to use real serial terminals

Maybe you have a glorious VT-100 terminal. To set terminals up for use by Linux, and thus making them available to the PiDP-8:

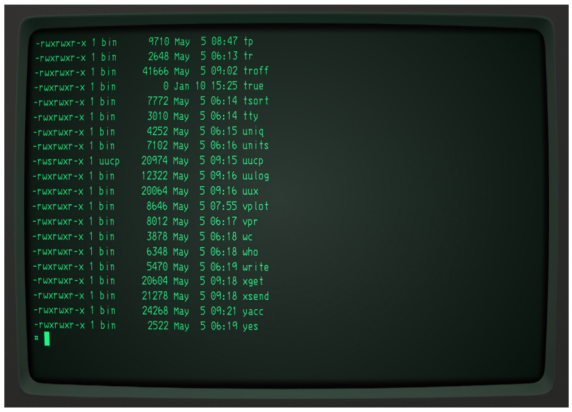
1. Copy the file `serial-getty@.service` from `/lib/systemd/system/` to `/etc/systemd/system/`

2. The file must be renamed as `serial-getty@ttyUSB0.service`, so that it points to your serial terminal which will be ttyUSB0.

3. The file may need to be edited with the parameters of your serial terminal, or the default setting may work. In my case I changed the first line in the [Service] section to read:
`ExecStart=-/sbin/agetty ttyUSB0 9600 vt100`

4. Now, enable the service for the adapter with: `sudo systemctl enable serial-getty@ttyUSB0.service`. You should get a 'Creating sim link ....' message. The service will now start up at every boot. Reboot.

A very nice CRT emulator



See the screen shot. There is a simulator for all sorts of CRT displays: cool-retro-term. It might improve life with your flat-panelled laptop.

Linux users can download it here: github.com/Swordfish90/cool-retro-term .

You might run it on your Linux laptop. You can, however, also run it on the Pi itself if you use a HDMI monitor or VNC. For that:

```
# Install the prerequisites:

sudo apt install build-essential qmlscene qt5-qmake qt5-default
qtdeclarative5-dev qml-module-qtquick-controls qml-module-
qtgraphicaleffects qml-module-qtquick-dialogs qml-module-qtquick-
localstorage qml-module-qtquick-window2 qml-module-qt-labs-settings qml-
module-qt-labs-folderlistmodel


# Get it from GitHub
git clone --recursive https://github.com/Swordfish90/cool-retro-term.git

# Build it
cd cool-retro-term

# Compile
qmake && make
```

Before starting the program (`./cool-retro-term`) you will need to enable the Pi's GPU using `sudo raspi-config` → `advanced options` → `GL Driver` → `GL Fake KMS`, and reboot. When in Cool itself, switch off a lot of the CPU intensive graphics options (noise, jitter) and performance becomes acceptable.

Note that the author of Cool asks for a small contribution if you like his work, see the github page.